

Popular Computing

Volume 8 Number 7

July 1980

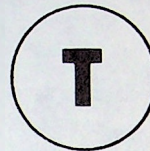
88

The world's only magazine devoted to the art of computing.

10	2	30	3	50	9
11	0	31	0	51	4
12	3	32	5	52	3
13	0	33	4	53	0
14	2	34	6	54	4
15	4	35	4	55	4
16	4	36	4	56	12
17	0	37	0	57	4
18	3	38	2	58	2
19	0	39	4	59	0
20	3	40	4	60	4
21	4	41	0	61	0
22	2	42	3	62	2
23	0	43	0	63	6
24	4	44	3	64	18
25	4	45	6	65	4
26	6	46	2	66	3
27	6	47	0	67	0
28	3	48	5	68	3
29	0	49	4	69	4

The Menomonee Falls Index

Index	First appearance (starting at N = 10)
2	14
3	18
4	15
5	32
6	26
7	128
8	81
9	50
10	118
11	5120
12	56
13	8192
14	1354
15	176
16	297
17	----
18	64
19	----
20	204
21	320
--	
--	
--	
99	7168



Some entries for a
table of results
for the
Menomonee Falls Index

Publisher: Audrey Gruenberger

Editor: Fred Gruenberger

Associate Editors: David Babcock
Irwin Greenwald
Patrick Hall

Contributing Editors: Richard Andree
William C. McGee
Thomas R. Parkin
Edward Ryan

Art Director: John G. Scott

Business Manager: Ben Moore

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is \$20.50 per year, or \$17.50 if remittance accompanies the order. For Canada and Mexico, add \$1.50 per year. For all other countries, add \$3.50 per year. Back issues \$2.50 each. Copyright 1980 by POPULAR COMPUTING.

@ 2023 This work is licensed under CC BY-NC-SA 4.0

The Menomonee Falls Index

In the table shown on the cover, each of the integers, N, from 10 through 69 is given an index number which is the product of the number of prime factors of N times its distance from the nearest prime number. Thus,

$$56 = 2 \cdot 2 \cdot 2 \cdot 7$$

has 4 factors and it is 3 away from the nearest prime (either 53 or 59). Similarly:

$$10816 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 13 \cdot 13$$

has 8 factors and is 15 away from the nearest prime (10831), so it has an index of 120.

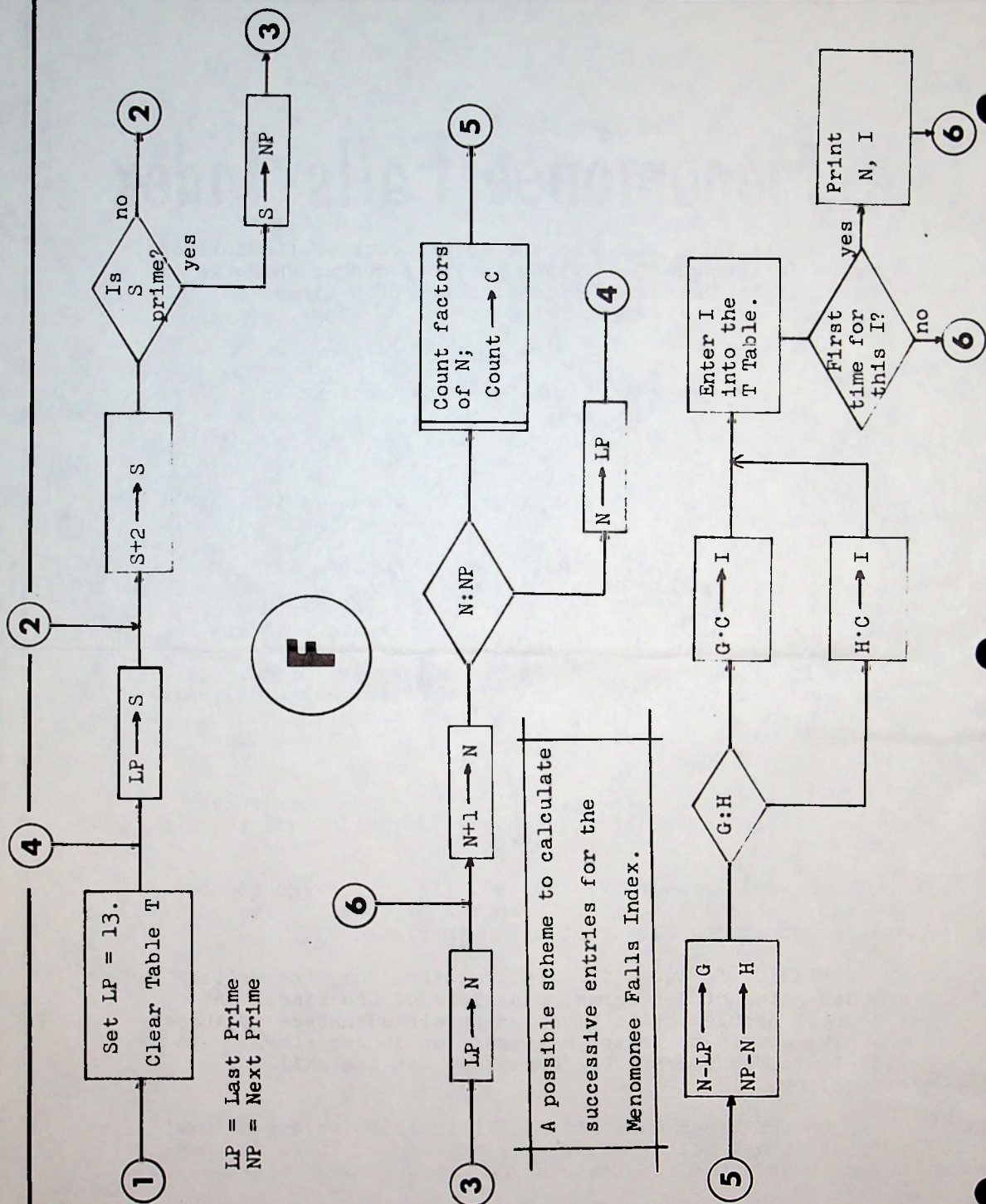
We note that all prime numbers will thus have an index of zero, and that it is not possible to have an index of one (at least for N greater than 10, where we chose to begin). The index numbers have no upper bound, inasmuch as the number of factors that a number can have can increase indefinitely, and the gaps between successive primes can also increase without bound. Even relatively small values of N can have a relatively large index number. Thus, N = 89728 has 8 factors and is 25 away from its nearest prime, for an index of 200.

If we tabulate the first appearance of each index number, we get a table like Table T, for which all entries from 2 through 99 certainly exist.

We would like to see Table T filled in, and to do so will certainly require a computer program. Its logic will be something like that shown in Figure F.

Filling in Table T, then, requires only the writing and debugging of a program, plus lots of CPU time. As a contest problem, this gives an unfair advantage to those who have access to faster machines for longer times. We wish to foster the art of computing, not the skill of embezzling.

On the other hand, it is difficult to select a "best" program in any sort of objective manner. There is something about a program that can be measured objectively, however.



So, for our Contest 18, we will award one prize of \$150 for the following:

1) Table T filled in for at least 24 more entries.

2) A program in BASIC to perform the calculations. This program must be written in some sort of "standard" BASIC such that it could run unchanged on several different machines, under different BASIC interpreters, including those of machines like Apple II, and TRS-80. In other words, the BASIC notation must avoid those special or extended features that are not common to many BASICs, but are unique to one particular system. "Portable BASIC"--that's what we seek.

3) Given items (1) and (2), the winner will be the program containing the smallest number of characters, not counting line numbers or insignificant spaces. Thus, the following statements:

```
1230 N = 5
1240 K = 6
```

count as 6 characters, while the single line:

```
1250 N = 5 : K = 6
```

counts as 7 characters. Note that the statement:

```
1260 PRINT "THIS IS THE RESULT"
```

contains 25 characters, since the spaces in the message are significant. (We recommend that your contest entry NOT contain such frills.)

It seems that we did not allow sufficient time in our earlier contests. For Contest 18, take all of 1980. Any entry must be received at:

Contest 18
POPULAR COMPUTING
Box 272
Calabasas, CA 91302

before New Year's Eve, December 31, 1980.

PROBLEM 275



Andree again

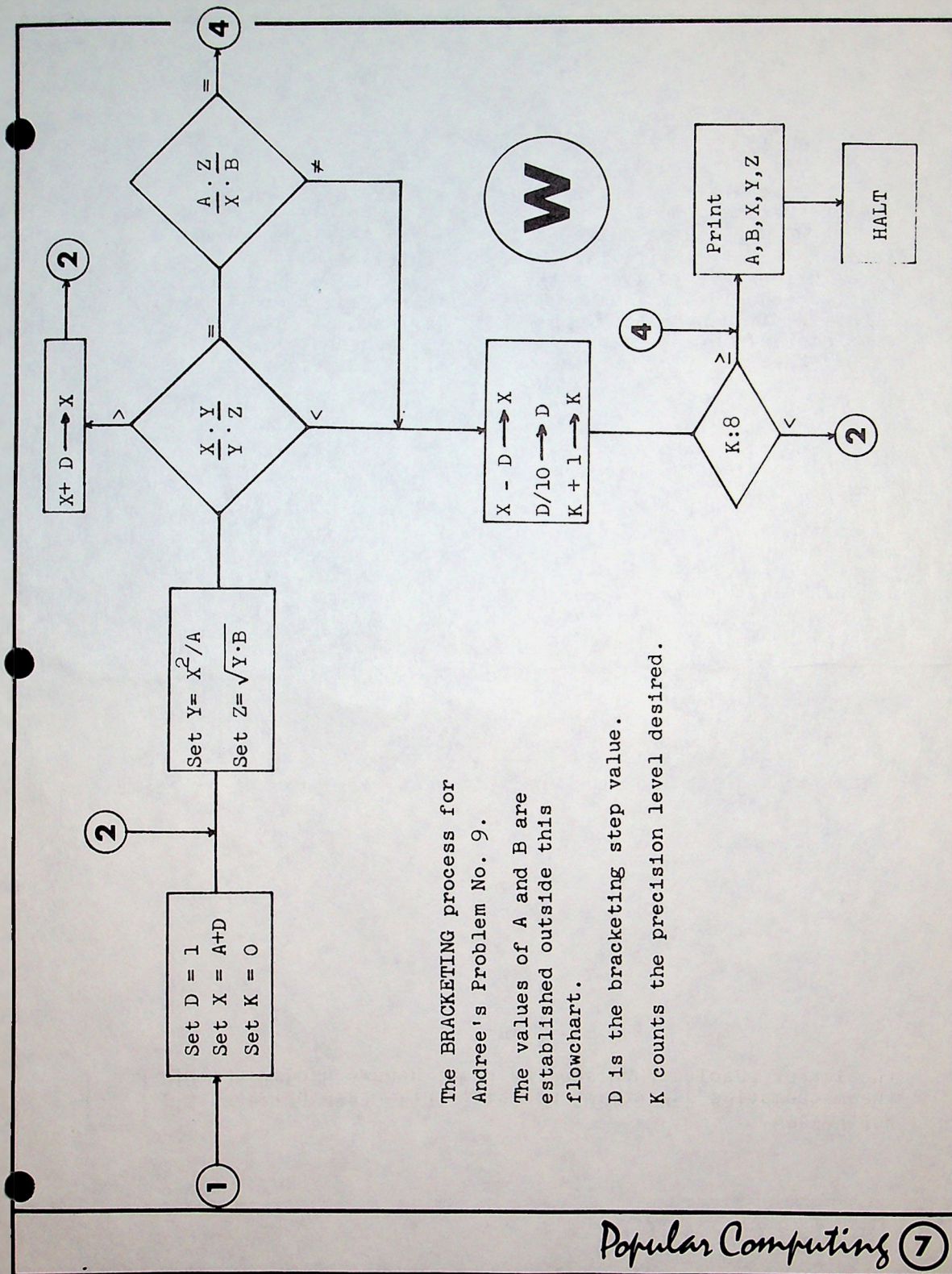
One of Professor Richard Andree's little training problems is this:

Given A and B, can we find X, Y, and Z
such that:

$$\frac{A}{X} = \frac{X}{Y} = \frac{Y}{Z} = \frac{Z}{B} ?$$

A nice problem. It has a closed-form solution (given at the end of this article), or the required values can be found by various iterative schemes, such as bracketing (the bracketing process has been discussed in our issues 35, 39, 44 and 48). To summarize the bracketing process:

1. The solution space must be defined; that is, lower and upper bounds on the required value must be established. This is vital--you must know the region in which the answer lies in order to apply bracketing intelligently.
 2. A sensible value, D, must be established, to use in stepping through that space.
 3. A test must be devised for determining when the required value is found, or when it has been passed.
 4. If the required value is found, quit. (This exit is generally rare.)
 5. If the required value has been passed, back up (at least*) one full step of D; cut the D value down (usually by a factor of ten); and continue stepping (step 2).
 6. Each passage through step 5 will increase the precision of the result by one decimal digit. Test for the required precision level and end the process when it is reached.
- (*) In some cases (e.g., searching for a minimum or a maximum) it is necessary to back off two full steps.



Flowchart W suggests a way to attack this problem. The bracketing process, like any other numerical algorithm, cannot be applied blindly. The process is simple-minded (and inelegant, inefficient, and unsophisticated) but straightforward. Its great virtues are these: it is easy to code in any language; it is fairly idiot-proof (that is, it has no subtle booby traps); and it fits an amazing range of problems. It is a quick-and-dirty method that is highly useful for one-time solutions.

The flowchart shows a way to bracket in on the three values required in Andree's problem. There is a tacit assumption that A and B are positive and distinct, and that B is greater than A and, indeed, sufficiently greater so that a starting value of A+D for X is within the solution space. It is a characteristic of the bracketing process that if one wanders outside the solution space, it is unlikely that the process will ever converge. The test (step 3 of the process) consists here of two tests, given in the large diamonds.

For a broad range of values of A and B, the logic of flowchart W, expressed in Applesoft BASIC, gave values of X, Y, and Z that differed by no more than one in the 9th significant digit from the analytic results:

$$X = \sqrt[4]{A^3 \cdot B}$$

$$Y = \sqrt{A \cdot B}$$

$$Z = \sqrt[4]{A \cdot B^3}$$

the latter results furnished by Prof. Robert Henderson, of the mathematics department at California State University, Northridge.



The Telephone Game



Rules: Any number can play. Each player starts at S and moves, in turn, toward F. Each play is dictated by the toss of three cubical dice, each of which bears the numbers 1-1-2-2-3-3 on its faces.

The dice are honest, and are to be tossed honestly. Additional moves backward and forward are indicated on the playing board on the following two pages.

Our apologies to our overseas friends who may not be able to connect up U.S. area codes with the cities.

Each of the three dice may be simulated in the following way. Assume the availability of a subroutine, RND, whose output, X, is uniformly distributed thus:

$$0 \leq X \leq 1$$

Then (using BASIC notation), the action of one die is given by:

$$X1 = \text{INT}(3 * \text{RND} + 1)$$

and the required action for the three dice is given by:

$$X = X1 + X2 + X3.$$

The distribution of the dice tosses should tend toward:

Total	3	4	5	6	7	8	9
Frequency	1	3	6	7	6	3	1

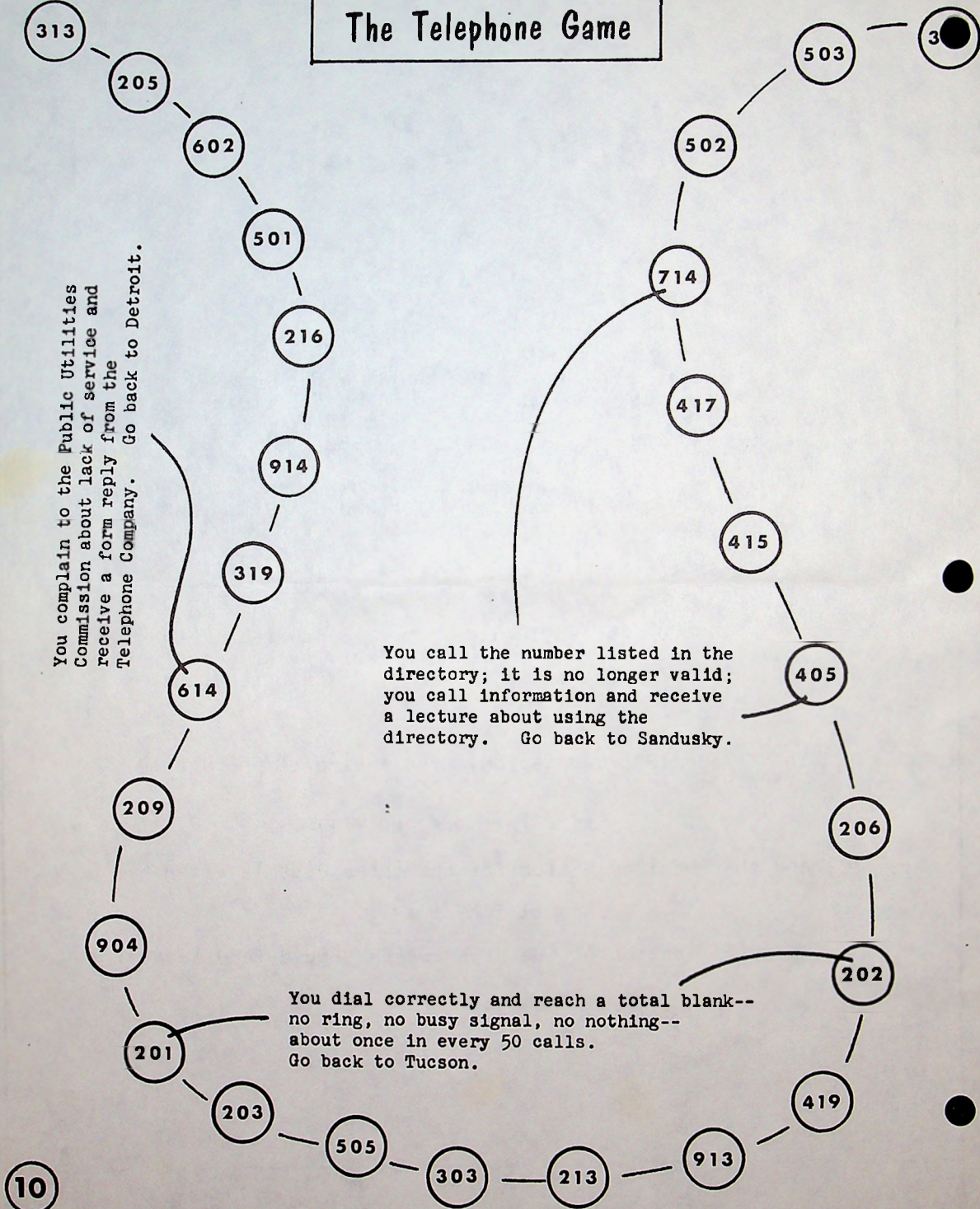
S

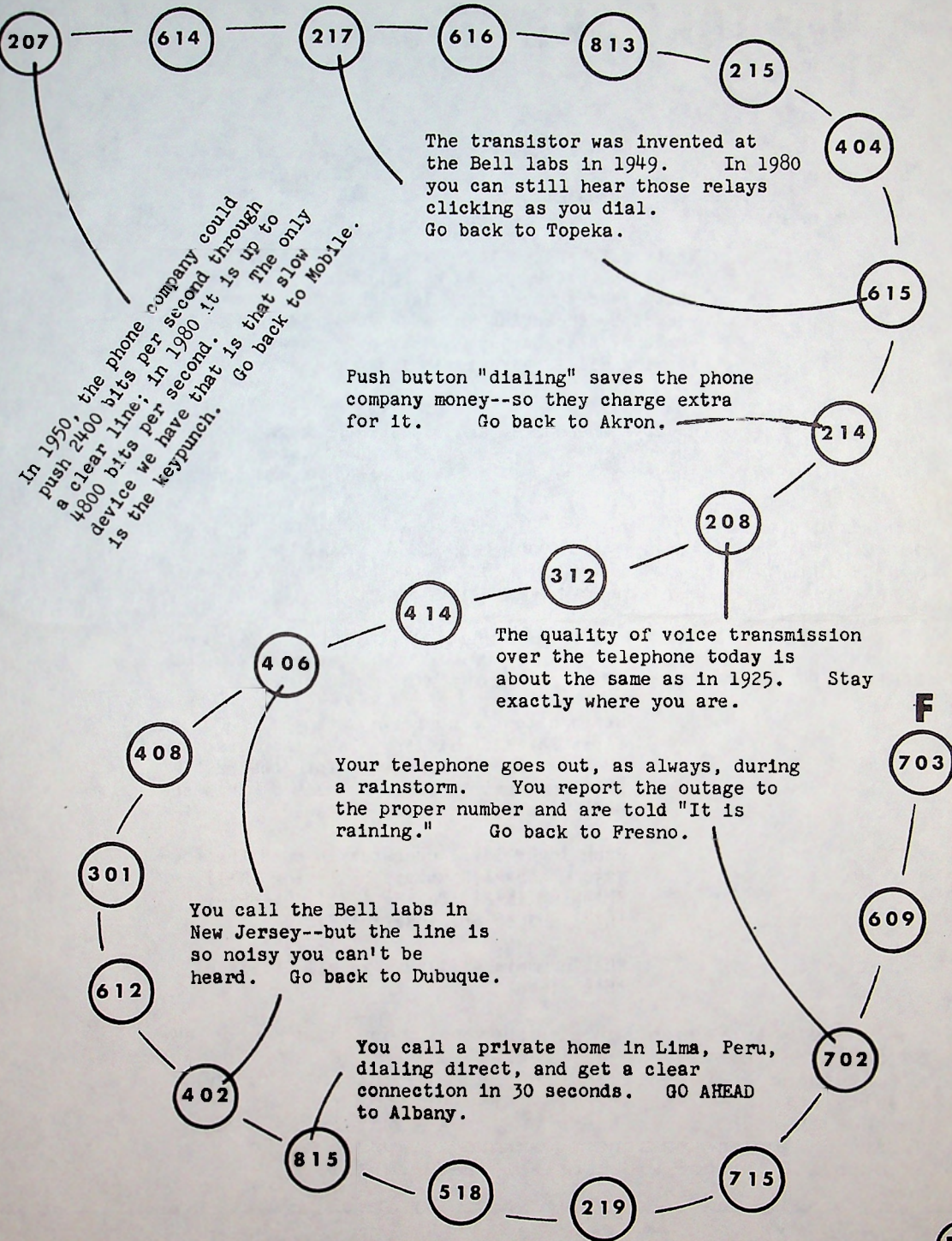
The Telephone Game

You complain to the Public Utilities Commission about lack of service and receive a form reply from the Telephone Company. Go back to Detroit.

You call the number listed in the directory; it is no longer valid; you call information and receive a lecture about using the directory. Go back to Sandusky.

You dial correctly and reach a total blank-- no ring, no busy signal, no nothing-- about once in every 50 calls. Go back to Tucson.





A New TAKE/SKIP Problem

Brief recapitulation: The original TAKE/SKIP problem first appeared as the K-Level Sieve Problem in our issue number 38; it was our Contest No. 7. The original problem was this:

Start with the positive integers. At each level, K, of the procedure to be followed, the numbers that are outputted from level K-1 are to be sieved by accepting K numbers and rejecting K numbers, alternately. What numbers will survive all levels of sieving?

The positive integers form level zero. Level 1 accepts one number and rejects the next; thus, the output of level 1 (which constitutes the input to level 2) is the sequence of odd numbers.

The winning solution appeared in issue 43. The central concept of the winning solution--a Bucket Brigade--was applied to the problem in flowchart form in issue 76, together with a possible program in BASIC.

We wish to reexamine the logic of the Bucket Brigade, applied to a new version of the problem. We'll call it TAKE/SKIP7 (and notice that the notation has been changed):

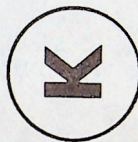
Start with the positive integers. For the first level of sieving, Take the first two numbers, Skip the next 3, Take the next 4, Skip the next 5, Take the next 6, Skip the next 7, and so on, indefinitely.

Each lower level operates in much the same way. Level K starts by Taking (K+1), Skipping (K+2), Taking (K+3), Skipping (K+4) and so on, indefinitely.

What numbers will survive all levels of this sieve?

The main reason for considering these sieve problems is that they provide ideal problem situations for those who are learning computing, and they come complete with known results. For TAKE/SKIP7 we have another reason; namely, to show the contrast between an implementation in BASIC versus an implementation in machine language. For such a contrast, integer problems are ideal examples.

<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>9</u>	<u>10</u>	<u>11</u>	<u>12</u>	<u>13</u>	<u>14</u>	<u>15</u>	<u>16</u>	<u>17</u>	<u>18</u>	<u>19</u>	<u>20</u>	<u>21</u>
<u>1</u>	<u>2</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>9</u>	<u>15</u>	<u>16</u>	<u>17</u>	<u>18</u>	<u>19</u>	<u>20</u>	<u>28</u>	<u>29</u>	<u>30</u>	<u>31</u>	<u>32</u>	<u>33</u>	<u>34</u>	<u>35</u>	
<u>1</u>	<u>2</u>	<u>6</u>	<u>16</u>	<u>17</u>	<u>18</u>	<u>19</u>	<u>20</u>	<u>34</u>	<u>35</u>	<u>45</u>	<u>46</u>	<u>47</u>	<u>48</u>	<u>49</u>	<u>69</u>	<u>70</u>	<u>71</u>	<u>72</u>		
<u>1</u>	<u>2</u>	<u>6</u>	<u>16</u>	<u>35</u>	<u>45</u>	<u>46</u>	<u>47</u>	<u>48</u>	<u>49</u>	<u>76</u>	<u>77</u>	<u>101</u>	<u>102</u>	<u>103</u>	<u>104</u>	<u>120</u>	<u>121</u>			
<u>1</u>	<u>2</u>	<u>6</u>	<u>16</u>	<u>35</u>	<u>77</u>	<u>101</u>	<u>102</u>	<u>103</u>	<u>104</u>	<u>120</u>	<u>121</u>	<u>168</u>	<u>202</u>							



Some intermediate results of this TAKE/SKIP sieve.

The lines above show the start of levels 2 through 6. The top line (level 2) is what is passed down from level 1, which is not shown.

Level 1 has for input all the natural numbers; it passes on to level 2 the numbers shown here on the top line.

At each level, the underscored numbers are those passed on to the next level. We can see here that the numbers 1, 2, 6, 16, 35, 77, and 202 will survive all levels of the sieve.

The problem could be attacked directly as indicated in Figure K. The natural numbers could be generated in a block of storage, and the process of accepting and rejecting could be applied to the numbers in the block. Figure K shows the start of the first 5 levels of the process, from which we can ascertain the first 6 numbers that will survive. Since the process eliminates half of the numbers at each level, then if we start with 33,000,000 numbers at level 1, we will have left only the first 6 numbers at level 25.

But the winners of our seventh contest (Tom Duff and Hugh Redelmeier, then graduate students at the University of Toronto) pointed out that the proper technique is the Bucket Brigade, for which the logic is outlined in the flowcharts R and S.

We have laid out the logic to work through 100 levels, which is far beyond any sensible limit. Flowchart R shows the mechanism for level 1, whose output is the top line of Figure K.

Flowchart S shows the logic for all the other levels. For each level, M, we need just three things: a flag, labelled T(M) to indicate whether the level is in the TAKE mode or the SKIP mode; a counter K(M) to count the number of numbers to be skipped or passed on to the next level; and a limit L(M) to test counter K(M) against. In the housekeeping phase of the problem, all the T's are set to zero (that is, every level is set initially to TAKE); all the K's are set to zero; and each L(M) is initialized to M+1.

Consider, for example, level 7, for which we begin with these values:

$$\begin{aligned} M &= 7 \\ K(7) &= 0 \\ T(7) &= 0 \\ L(7) &= 8 \end{aligned}$$

The first 8 numbers that are received from level 6 will be passed on to level 8, after which T(7) is changed to one. The next 9 numbers are to be rejected, and for each of them, control returns immediately to level 1, the MAIN program. Figure P shows the first 24 results.

1
 2
 6
 16
 35
 77
 202
 448
 967
 2 330
 4 993
 10 980
 23 350
 48 721
 104 029
 216 238
 453 241
 924 603

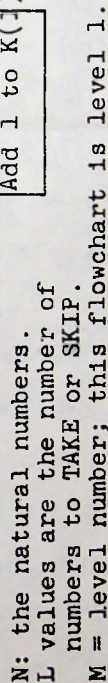
 2 000 448
 4 239 397
 9 056 677
 19 972 068
 41 026 182
 82 789 278

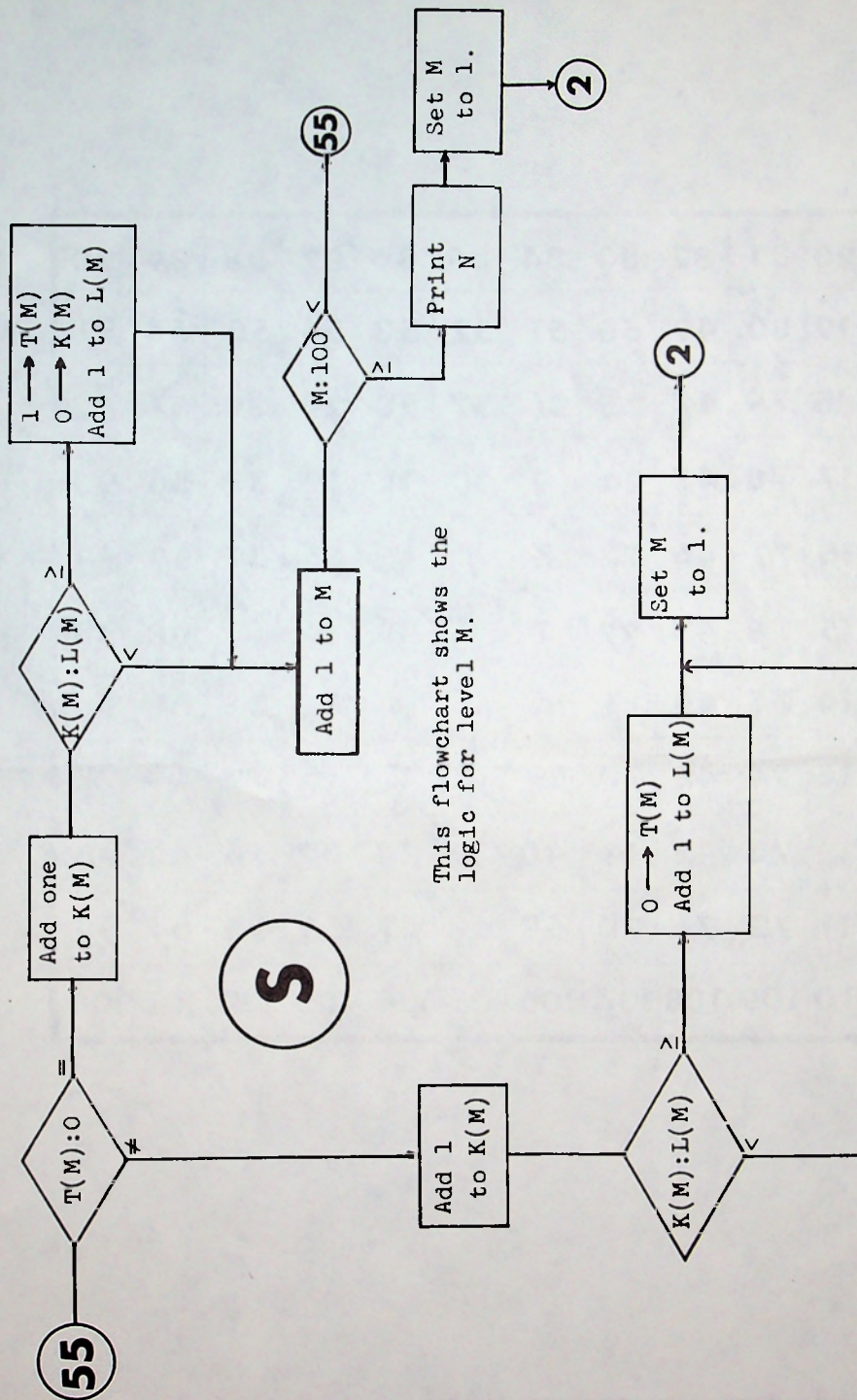
The first 24 numbers from the TAKE/SKIP7 sieve. The first 18 of these were found with a program in BASIC (Applesoft) in 13 hours and 40 minutes. The identical logic was recoded for 6502 machine language, and that program obtained the same 18 results in exactly 4 minutes. The ratio of these speeds is 205 to 1.

The time for the machine-language program to obtain the first 24 results was 5 hours, 56 minutes (roughly, each result takes twice as long to calculate as the previous one.)

P

T values are flags for $\begin{cases} 0 = \text{TAKE} \\ 1 = \text{SKIP} \end{cases}$



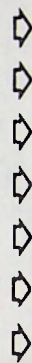


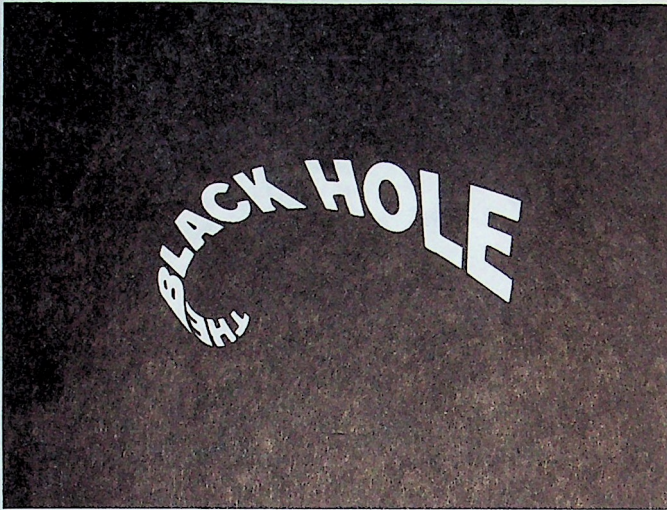
This flowchart shows the logic for level M.

S



120	81	82	83	84	85	86	87	88	89	90
119	80	49	50	51	52	53	54	55	56	91
118	79	48	25	26	27	28	29	30	57	92
117	78	47	24	9	10	11	12	31	58	93
116	77	46	23	8	1	2	13	32	59	94
115	76	45	22	7		3	14	33	60	95
114	75	44	21	6	5	4	15	34	61	96
113	74	43	20	19	18	17	16	35	62	97
112	73	42	41	40	39	38	37	36	63	98
111	72	71	70	69	68	67	66	65	64	99
110	109	108	107	106	105	104	103	102	101	100





The numbers from 1 to 120 are arranged in a spiral around the Black Hole, as shown in the accompanying Figure.

The Black Hole annihilates any combination of not more than three numbers that form a multiple of 19. Thus, the Black Hole will extinguish combinations of numbers like:

$$35 + 41 \quad (= 4 \text{ times } 19)$$

$$43 \cdot 65 - 2 \quad (= 147 \text{ times } 19)$$

$$3^5 + 4 \quad (= 13 \text{ times } 19)$$

It is easy to establish that all of the 120 numbers can be swallowed up in groups of one, two, or three numbers that make up, in some arithmetic combination, a multiple of 19. There are an enormous number of ways that this can be done.

But now we add a natural constraint; namely, that the numbers must be able to be moved to the Black Hole in the fashion of square tiles in the plane. Thus, to start, only the numbers from 1 to 8 are eligible to be considered. At that, it is surprising in how many ways a multiple of 19 can be formed out of just those numbers, taking them three at a time:

$8 \cdot 2 + 3$	$8^6 - 1$
$6 \cdot 5 + 8$	$7^3 - 1$
$4 \cdot 5 - 1$	$8^2 - 7$
$8 + 7 + 4$	$4^3 - 7$
$8 + 6 + 5$	$5^2 - 6$
$8 \cdot 7 + 1$	$7 \cdot 2 + 5$
$8 \cdot 5 - 2$	$7 \cdot 3 - 2$
$8 \cdot 4 + 6$	$7 \cdot 5 + 3$
$8 \cdot 3 - 5$	$7 \cdot 6 - 4$

But our constraint is tighter than that. The triple:

$$8 \cdot 4 + 6 \quad (= 2 \text{ times } 19)$$

will not do for a starter, since none of those three numbers can move to the Black Hole before numbers 3, 5, or 7 move first. A triple like:

$$7 \cdot 5 + 3 \quad (2 \text{ times } 19)$$

is fine for a starter, after which all those numbers:

1, 2, 8, 4, 6, 22, 18, 14

could then be used. In fact, after $7 \cdot 5 + 3$, the next triple could be $13 + 14 - 8$, since the 13 can slide out after the 14.

The object is to pour all the 120 numbers into the Black Hole, in groups of one, two, or three numbers at a time, forming multiples of 19 in any way, subject to the stated constraint.

We have our usual questions:

- A) Can it be done?
- B) Is it a computer problem?

